



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/657,468	09/08/2003	Makarand Gadre	MS1-1597US	9819
22801	7590	01/09/2008		
LEE & HAYES PLLC 421 W RIVERSIDE AVENUE SUITE 500 SPOKANE, WA 99201			EXAMINER STEELMAN, MARY J	
			ART UNIT 2191	PAPER NUMBER
			MAIL DATE 01/09/2008	DELIVERY MODE PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary

Application No.

10/657,468

Applicant(s)

GADRE ET AL.

Examiner

MARY STEELMAN

Art Unit

2191

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 16 October 2007.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1,2,4-12,14-17 and 19-28 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☐ Claim(s) 1,2,4-12,14-17 and 19-28 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/ are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
 - ☐ Certified copies of the priority documents have been received in Application No. _____.
 - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) ☐ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☐ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____.

- 4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____.
- 5) ☐ Notice of Informal Patent Application
- 6) ☐ Other: _____.

DETAILED ACTION

1. This Office Action is in response to RCE, Remarks and Amendments received 10/16/2007. Per Applicant's request, claims 1, 12, 17, 19, 20, and 23 have been amended. Claims 1, 2, 4-12, 14-17, and 19-28 are pending.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned

with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 1, 12, 17, and 23 are provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 1, 13, and 25 of copending Application No. 10/657463. Although the conflicting claims are not identical, they are not patentably distinct from each other because claims recite generating common intermediate language, receiving a portion of JAVA language source code / object oriented language source code that compiles source code into non-language-neutral bytecodes, referencing a first class having a definition that is uniformly applicable to a plurality of classes (generic class), the source code identifying one of the plurality of associated classes (compiling an instance of the generic class) and generating language neutral intermediate language (common intermediate language code). Although the claims are not using the identical terminology, they are addressing similar elements.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Examiner maintains the double patenting rejection. Application 10/657463, claim 1 recites a JAVA language source code interoperating with a language other than the JAVA language .NET), generating language neutral intermediate language (CIL).

Response to Arguments

4. Applicant has argued, in substance, the following:

(A) As noted on page 12 of Remarks, claim 1 is amended to clarify that "the first object oriented language source code being associated with a framework developed for use with a predetermined object oriented programming language that compiles source code into non-language-neutral bytecodes" and that the second object oriented source code is different that the first object oriented source code.

Examiner's Response:

It appears that the Kennedy reference reads on the amended claim language. Kennedy disclosed (p. 1, lower left column) "The CLR (runtime engine)...providing a common type system and intermediate language for executing program written in a variety of languages (a second object oriented source code different...), and for facilitating inter-operability between those languages..." Page 2, lower right column, "In this section we show how our support for parametric polymorphism in the CLR allows a generics mechanism to be added to the language C# with relative ease." Examiner maintains that Kennedy disclosed the limitations of claim 1.

(B) As noted on page 13, last paragraph, Kennedy has not been shown to disclose or suggest 'writing first object oriented language source code...associated with a framework developed for

use with a predetermined object oriented programming language that compiles source code into non-language-neutral bytecodes” and “receiving a second different object oriented language source code referencing the generic class defined by the first object oriented language source code.”

Examiner's Response:

Examiner disagrees. Kennedy disclosed “inter-operability” between languages (p. 1, lower left column), an intermediate language and runtime engine (EE) (page 1, lower right column), support for parametric polymorphism in the CLR allows generics mechanism (p. 2, lower right column), intermediate language of the CLR, called IL (p. 4, lower right column). Kennedy concludes (p. 11, lower left column), “This paper has described the design and implementation of support for parametric polymorphism in the CLR. The system we have chosen is very expressive, and this means that it should provide a suitable basis for language inter-operability over polymorphic code.”

(C) As noted on page 14, 3rd paragraph, Applicant cites that independent claim 12, 17, and 23 are similar to claim 1.

Examiner's Response:

Examiner disagrees. Independent claim 12 recites (1) an existing object oriented language source code, and (2) a second source in a second source framework. Claim limitations fail to

recite that the second source is a different object oriented programming language, as claimed in claim 1.

Independent claim 17 is a system claim that recites (1) a first object oriented language source code, and (2) a second source application (is this a different object oriented programming language?). Claim 17 recites "wherein the means for receiving comprises a computer readable medium having stored thereon a second source application." What is the purpose of this second source application? Is it merely stored there, with no interaction? Claim limitations fail to recite that the second source is a different object oriented programming language, and that the second, different object oriented language source code references the generic class defined by the first object oriented language source code, as claimed in claim 1.

Examiner agrees that the claim limitations of independent claim 23, recite similar features as claim 1.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

6. Claims 1-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 6,760,905 B1 to Hostetter et al., in view of US Patent 6,018,628 to Stoutamire, and further in view of "Design and Implementation of Generics for the .NET common Language Runtime", by Andrew Kennedy and Don Syme (June 2001) (hereinafter Kennedy).

Per claim 1:

A method of generating common intermediate language code comprising:

- writing first object oriented language source code that comprises a definition of a generic class usable in a framework, the first object oriented language source code being associated with a framework developed for use with a predetermined object oriented programming language that compiles source code into non-language-neutral bytecodes;
- generating an instance of the generic class;
- compiling the instance of the generic class into common intermediate language code executable by a runtime engine.

Hostetter disclosed (col. 2:62-67) JAVA programming language with the usage of parameters within the source code definition of a class template, compilation, and (col. 3:4-15) creating a class from the class template (generating an instance of the generic class). Col. 5:58-col. 6: 9, Source code defining the class template is compiled into an object representation suitable for use as a resource when subsequently compiling classes based on the template... These source code representations may be some type of intermediate code (i.e. code categorized between source code and object code)... In general, a source code representation is an abstract representation

(common intermediate language code) of the source code of the class template not yet specialized for any particular template generated class.

Stoutamire more (Abstract), explicitly disclosed generating code using parameterized classes, to generate unparameterized class code (an instance of the generic class), (col. 12: 27) generating byte code from parameterized class files. Col. 14: 64, by adding certain annotations of flags to the byte code, modified virtual machines can take further advantage. Col. 15: 30, certain embodiments have been described using the JAVA programming language, the present invention can be practiced on a variety of programming languages...

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention to modify Hostetter, using the teachings of Stoutamire, to produce an invention that efficiently allocates memory (Hostetter, col. 2:65-col. 3: 3, Stoutamire, col. 2:48-64) by using parameterized classes and generating only necessary class instances.

Hostetter / Stoutamire failed to disclose:

-receiving a second, different object oriented language source code referencing the generic class defined by the first object oriented language source code.

However, Kennedy disclosed (page 1, left column, last paragraph), "The CLR has the ambitious aim of providing a common type system and intermediate language for executing programs written in a variety of languages, and for facilitating inter-operability between those languages.

Kennedy disclosed “inter-operability” between languages (p. 1, lower left column), an intermediate language and runtime engine (EE) (page 1, lower right column), support for parametric polymorphism in the CLR allows generics mechanism (p. 2, lower right column), intermediate language of the CLR, called IL (p. 4, lower right column). Kennedy concludes (p. 11, lower left column), “This paper has described the design and implementation of support for parametric polymorphism in the CLR. The system we have chosen is very expressive, and this means that it should provide a **suitable basis for language inter-operaility over polymorphic code.**”

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Hostetter / Stoutamire, using the teachings of Kennedy, because (Kennedy, page 1, left column, last sentence) it “relieves programmers of the burden of describing the data marshalling (typically through an interface definition language, or IDL) that is necessary for language interoperation. Note that Kennedy disclosed polymorphic methods and parameterized types / parameterized classes (page 4).

Per claim 2:

-storing the source code in a class library of the framework.

Hostetter : Col. 5: 63, “resource of source code for compiling’ Col. 6: 19-20, “source code declaring a class based on the class template is compiled”

Per claim 4:

-parsing the second source code into a parse tree representing the second source code.

Hostetter : Col. 3: 66 – col. 4: 7, “The process for generating method bindings is triggered whenever a method call instruction, referencing a class method of a template-generated class, requires compilation. It involves (1) scanning the class representation...(2) creating a method binding; (3) compiling (create parse tree) the source code representation.” Col. 4: 27, “The runtime compiler is invoked ...

Regarding the ‘second source code’ being a different type of source code language, Kennedy disclosed (page 1, left column, last paragraph), “The CLR has the ambitious aim of providing a common type system and intermediate language for executing programs written in a variety of languages, and for facilitating inter-operability between those languages.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Hostetter / Stoutamire, using the teachings of Kennedy, because (Kennedy, page 1, left column, last sentence) it “relieves programmers of the burden of describing the data marshalling (typically through an interface definition language, or IDL) that is necessary for language interoperation. Note that Kennedy disclosed polymorphic methods and parameterized types / parameterized classes (page 4).

Per claim 5:

-parsing the portion of the object oriented language source code into a parse tree representing the

source code.

See rejection of limitations addressed in claim 4 above.

Per claim 6:

-writing first object oriented language source code comprises defining at least one parameter associated with the generic class.

Hostetter : Col. 12:50, class object is defined by its parameters

Per claim 7:

-the at least one parameter is an unconstrained type.

Hostetter : Col. 12: 50-58

Per claim 8:

-declaring an instance of the generic class in second object oriented source code.

Hostetter : Col. 6:50-53, the source code representation that constitutes the body of a method is compiled when the program executes a compiled method call instruction invoking that method.

The first time stub code is referenced which initiates compilation of a generic class.

Regarding a second object oriented source code that is in a different source code language, Kennedy disclosed (page 1, left column, last paragraph), "The CLR has the ambitions aim of providing a common type system and intermediate language for executing programs written in a variety of languages, and for facilitating inter-operability between those languages.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Hostetter / Stoutamire, using the teachings of Kennedy, because (Kennedy, page 1, left column, last sentence) it “relieves programmers of the burden of describing the data marshalling (typically through an interface definition language, or IDL) that is necessary for language interoperation. Note that Kennedy disclosed polymorphic methods and parameterized types / parameterized classes (page 4).

Per claim 9:

-declaring an instance of the generic class comprises specifying a type from a plurality of allowable types associated with the generic class.

Hostetter : Col. 8: 34-41, template generated classes used in program code are data types, declared by a variable declaration statement. Parameters of the class template are replaced with actual data types.

Per claim 10:

-the specified type is another generic class.

Hostetter : Col. 8: 34-41 & col. 7: 23, ‘inherited classes’

Per claim 11:

-the generic class comprises one of: a Queue class; a Dictionary class; and a Stack class.

Hostetter : As an example, col. 9:1-5, ClassObject stores binding objects as elements of a linked list (queue class).

Per claim 12:

A method of using a generic class comprising:

- adapting existing object oriented language source code, the existing object oriented language source code being associated with a framework developed for use with a predetermined object oriented programming language that compiles source code into non-language-neutral bytecodes, to include a declaration of a first generic class provided by a software package having a class definition of the first generic class;
- compiling the adapted object oriented language source code with the class definition to generate common intermediate language code.

See rejection of claim 1 above. Hostetter : Col. 8: 24, generating object representations of template generated classes. Col. 8: 42, Compiler encounters declared type as template generated class, and class is compiled (compiling source code with class definition) into a template generated class representation.

Stoutamire more (Abstract), explicitly disclosed generating code using parameterized classes, to generate unparameterized class code (an instance of the generic class), (col. 12: 27) generating byte code from parameterized class files. Col. 14: 64, by adding certain annotations of flags to the byte code, modified virtual machines can take further advantage . Col. 15: 30, certain

embodiments have been described using the JAVA programming language, the present invention can be practiced on a variety of programming languages...

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention to modify Hostetter, using the teachings of Stoutamire, to produce an invention that efficiently allocates memory (Hostetter, col. 2:65-col. 3: 3, Stoutamire, col. 2:48-64) by using parameterized classes and generating only necessary class instances.

Hostetter / Stoutamire failed to disclose:

-wherein the adapting comprises editing the existing object oriented language source code with a second source in a second source framework;

However, Kennedy disclosed (page 1, left column, last paragraph), "The CLR has the ambitious aim of providing a common type system and intermediate language for executing programs written in a variety of languages, and for facilitating inter-operability between those languages. Note that the CLR (p. 2, middle right column) does a 'just in time' type of compilation.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Hostetter / Stoutamire, using the teachings of Kennedy, because (Kennedy, page 1, left column, last sentence) it "relieves programmers of the burden of describing the data marshalling (typically through an interface definition language, or IDL) that is necessary for

language interoperation. Note that Kennedy disclosed polymorphic methods and parameterized types / parameterized classes (page 4).

Per claim 14:

-the class definition defines at least one parameter of the generic class.

Hostetter : Col. 8: 34-42.

Per claim 15:

-compiling comprises: validating a specified type of the generic class according to the class definition.

Hostetter : Col. 6: 29-30, col. 9: 21-30, type object defines the field in terms of what values it can store, col. 13:1-15.

Per claim 16:

-adapting comprises nesting a second generic class in the declaration of the first generic class.

Hostetter : Col. 11: 34, class templates are inherited (nesting)

Per claim 17:

A system for authoring source code comprising:

-a class library having a generic class definition;

-a means for receiving a declaration of an instance of the generic class in first object oriented language source code

Hostetter : Col. 7: 32-35, field descriptors and method descriptors stored in ClassTemplate, Col. 8: 25, ClassTemplate provides source code representation (class library having a generic class definition) for all the methods and fields defined in the class template. Col. 8: 42-46, Compiler encounters variable declaration in which the declared type is a template generated class, the class is compiled into a template generated class representation, an object representation of the template generated class. Col. 9: 65, a method binding is created for each class method that is referenced by a source code instruction. Col. 10: 39-42, compiler compiles to a type signature object (instance of the generic class)

Stoutamire more (Abstract), explicitly disclosed generating code using parameterized classes, to generate unparameterized class code (an instance of the generic class), (col. 12: 27) generating byte code from parameterized class files. Col. 14: 64, by adding certain annotations of flags to the byte code, modified virtual machines can take further advantage . Col. 15: 30, certain embodiments have been described using the JAVA programming language, the present invention can be practices on a variety of programming languages...

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention to modify Hostetter, using the teachings of Stoutamire, to produce an invention that efficiently allocates memory (Hostetter, col. 2:65-col. 3: 3, Stoutamire, col. 2:48-64) by using parameterized classes and generating only necessary class instances.

Regarding the limitations:

-wherein the means for receiving comprises a computer readable medium having stored thereon a second source application; and

Kennedy disclosed interoperability of languages (a plurality of source code languages) (page 2, right column, 2nd paragraph).

-a means for generating metadata descriptive of the generic class.

Kennedy disclosed (page 11, bottom of left column & top of right column), “the dictionary passing scheme...i.e., by lazily creating dictionaries that record the essential information (metadata) that records how a type satisfies a constraint. See Kennedy, page 9, 4.4.1 – Pre-computing dictionaries of type handles.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Hostetter / Stoutamire, using the teachings of Kennedy, because (Kennedy, page 1, left column, last sentence) it “relieves programmers of the burden of describing the data marshalling (typically through an interface definition language, or IDL) that is necessary for language interoperation. Note that Kennedy disclosed polymorphic methods and parameterized types / parameterized classes (page 4).

Per claim 19:

-a common intermediate language importer operable to associate the generic class declaration in the first object oriented language source code to the generic class definition.

Hostetter : Col. 10: 15-39, associates generic class declarations.

Per claim 20:

-a semantic analyzer operable to validate the generic class declaration in the first object oriented language source code according to the generic class definition.

Hostetter : Col. 6:30-compiler bindings, col. 10: 32-36, check if method descriptors correspond to the referenced class method.

Per claim 21:

-a code generator operable to generate common intermediate language code representative of the generic class.

Hostetter : Col. 6: 29-30, first compilation results in the type signature of a method being compiled into a method binding. Col. 6: 24, class template representation (intermediate language)

Hostetter disclosed (col. 2:62-67) JAVA programming language with the usage of parameters within the source code definition of a class template, compilation, and (col. 3:4-15) creating a class from the class template (generating an instance of the generic class). Col. 5:58-col. 6: 9, Source code defining the class template is compiled into an object representation suitable for use as a resource when subsequently compiling classes based on the template...These source code representations may be some type of intermediate code (i.e. code categorized between source code and object code)...In general, a source code representation is an abstract representation (common intermediate language code) of the source code of the class template not yet specialized for any particular template generated class.

Stoutamire more (Abstract), explicitly disclosed generating code using parameterized classes, to generate unparameterized class code (an instance of the generic class), (col. 12: 27) generating byte code from parameterized class files. Col. 14: 64, by adding certain annotations of flags to the byte code, modified virtual machines can take further advantage. Col. 15: 30, certain embodiments have been described using the JAVA programming language, the present invention can be practiced on a variety of programming languages...

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention to modify Hostetter, using the teachings of Stoutamire, to produce an invention that efficiently allocates memory (Hostetter, col. 2:65-col. 3: 3, Stoutamire, col. 2:48-64) by using parameterized classes and generating only necessary class instances.

Per claim 22:

-a runtime engine operable to translate the common intermediate language into machine-specific binary executable by a computer associated with the runtime engine.

Hostetter : Col. 6: 33, resulting executable code

Per claim 23:

A computer-readable medium having stored thereon microprocessor-executable instructions for performing a method comprising:

-receiving input representing a generic class definition in a first object oriented language, the first object oriented language being associated with a framework developed for use with a

predetermined object oriented programming language that compiles source code into non-language-neutral bytecodes;

- receiving source code that references the generic class;

- compiling the source code with an instance of the generic class into common intermediate language code executable by a runtime engine.

- receiving a second, different object oriented language source code referencing the generic class defined by the first object oriented language source code.

See rejection of limitations addressed in claim 1 above.

Stoutamire more (Abstract), explicitly disclosed generating code using parameterized classes, to generate unparameterized class code (an instance of the generic class), (col. 12: 27) generating byte code from parameterized class files. Col. 14: 64, by adding certain annotations of flags to the byte code, modified virtual machines can take further advantage. Col. 15: 30, certain embodiments have been described using the JAVA programming language, the present invention can be practiced on a variety of programming languages...

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention to modify Hostetter, using the teachings of Stoutamire, to produce an invention that efficiently allocates memory (Hostetter, col. 2:65-col. 3: 3, Stoutamire, col. 2:48-64) by using parameterized classes and generating only necessary class instances.

However, Kennedy disclosed (page 1, left column, last paragraph), “The CLR has the ambitions aim of providing a common type system and intermediate language for executing programs written in a variety of languages, and for facilitating inter-operability between those languages.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Hostetter / Stoutamire, using the teachings of Kennedy, because (Kennedy, page 1, left column, last sentence) it “relieves programmers of the burden of describing the data marshalling (typically through an interface definition language, or IDL) that is necessary for language interoperation. Note that Kennedy disclosed polymorphic methods and parameterized types / parameterized classes (page 4).

Per claim 24:

-storing the generic class definition in a framework class library.

Hostetter : Col. 5: 61, class template representation

Per claim 25:

-the source code comprises object oriented language source code.

Hostetter : Col. 2: 63, col. 13: 65 source files written in a language that creates the output

Per claim 26:

-the method further comprises generating metadata describing the generic class.

See rejection of limitations addressed in claim 21 above.

Per claim 27:

-the generic class definition comprises a generic class name and two angular brackets around one or more parametric types.

Hostetter failed to explicitly disclose a generic class name and two angular brackets.

However, Hostetter disclosed, (col. 6: 5) a source code representation is an abstract representation of the source code of the class template not yet specialized...the source code representations stored in these method descriptors represent a type signature and a method body...the type signature represents the arguments that the method expects and the value that the method returns. The method body is the source code constituting the functionality. A method table is generated with entries for each defined method.

Kennedy provided an explicit example at page 3, left column: Class Stack<T>,

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Hostetter / Stoutamire, using the teachings of Kennedy, because (Kennedy, page 1, left column, last sentence) it “relieves programmers of the burden of describing the data marshalling (typically through an interface definition language, or IDL) that is necessary for language interoperation. Note that Kennedy disclosed polymorphic methods and parameterized types / parameterized classes (page 4).

Per claim 28:

-the method further comprises compiling the generic class definition into common intermediate language code.

Hostetter : Col. 6:1-61, Abstract, intermediate code is produced. At runtime, when a class method is invoked for the first time, stub code for that method initiates further compilation of a non-generic resulting executable object.

Conclusion

6.. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (571) 272-3704. The examiner can normally be reached Monday through Thursday, from 7:00 AM to 5:30 PM. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached at (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned: 571-273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR

Application/Control Number:
10/657,468
Art Unit: 2191

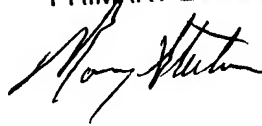
Page 24

system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Mary Steelman

01/06/2008

MARY STEELMAN
PRIMARY EXAMINER

A handwritten signature in black ink, appearing to read "Mary Steelman", written over the printed name and title.